8-11-2005

Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture

Yan Wang  
*University of Central Florida*

Bart O. Nnaji  
*University of Pittsburgh*

---

Follow this and additional works at: [http://lib.dr.iastate.edu/edesign_pubs](http://lib.dr.iastate.edu/edesign_pubs)  
Part of the [Computer-Aided Engineering and Design Commons](http://lib.dr.iastate.edu/edesign_pubs)

---

**Recommended Citation**  
*Center for e-Design Publications*. 6.  
[http://lib.dr.iastate.edu/edesign_pubs/6](http://lib.dr.iastate.edu/edesign_pubs/6)

---

This Article is brought to you for free and open access by the Center for e-Design at Iowa State University Digital Repository. It has been accepted for inclusion in Center for e-Design Publications by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture

Abstract
Current computer-aided design (CAD) systems only support interactive geometry generation, which is not ideal for distributed engineering services in enterprise-to-enterprise collaboration with a generic thin-client service-oriented architecture. This paper proposes a new feature-based modeling mechanism—document-driven design—to enable batch mode geometry construction for distributed CAD systems. A semantic feature model is developed to represent informative and communicative design intent. Feature semantics is explicitly captured as a trinary relation, which provides good extensibility and prevents semantics loss. Data interoperability between domains is enhanced by schema mapping and multiresolution semantics. This mechanism aims to enable asynchronous communication in distributed CAD environments with ease of design alternative evaluation and reuse, reduced human errors, and improved system throughput and utilization.

Keywords
feature-based modeling, form feature, semantic model, RDF/XML, interoperability, service-oriented architecture, collaborative design

Disciplines
Computer-Aided Engineering and Design

Comments
This article is from Journal of Computing and Information Science in Engineering 6 (2005): 127, doi:10.1115/1.2194911. Posted with permission.

This article is available at Iowa State University Digital Repository: http://lib.dr.iastate.edu/edesign_pubs/6
Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture

Yan Wang
Assistant Professor
NSF Center for e-Design,
University of Central Florida,
4000 Central Florida Blvd.,
Orlando, FL 32816-2993
e-mail: wangyan@mail.ucf.edu

Bart O. Nnaji
William Kepler Whiteford Professor
Center for e-Design,
University of Pittsburgh,
1046 Benedum Hall,
Pittsburgh, PA 15261-2210

1 Introduction

With the recent occurrence of outsourcing, collaborative product development among designers, manufacturers, suppliers, vendors, and other stakeholders is one of the keys for manufacturers to improve product quality, reduce cost, and shorten time-to-market in today’s global competition. Collaborative design is the new design process where multidisciplinary stakeholders participate in design decision making and share product information across enterprise boundaries in an Internet-enabled distributed environment.

Compared to traditional stand-alone computer-aided design (CAD) systems, there are new issues that need to be resolved in collaborative CAD. For example,

(i) Distributed geometric model: Current data models including STEP were designed for standalone CAD systems. Distributed databases need distributed data modeling schemes to optimize data access time and storage space.

(ii) Consistency management and version control: Design data are modified by multiple designers. Most recent and correct version should be maintained in either centralized or distributed repository.

(iii) Intellectual property protection: Collaborative design requires design data to be shared by different parties. Data security is essential to build trustworthy distributed CAD systems.

(iv) Model compression: Domain specific design data compression can improve communication performance given limited bandwidth and storage space.

Usually software systems may run in two modes: interactive mode, in which commands are entered and executed one at a time, and batch mode, in which commands are listed in a batch file sequentially and execution of the batch file finishes all commands automatically without user interaction. The issue of batch mode geometry generation for distributed CAD is discussed in this paper. Current CAD systems only support interactive geometry generation. CAD users create a geometric model by defining features step by step. These CAD systems can become fat clients in a distributed CAD environment, in which clients perform the bulk of data processing operations locally. However, in a simple web-based environment, thin-client CAD tools mainly with visualization functions cannot perform complex editing tasks locally. The majority of data processing requests are sent to the server. Synchronous communication will become the bottleneck of the overall system performance. Thus, synchronous and interactive model generation is not ideal for a distributed CAD system in which a thin-client infrastructure is used in regular enterprise-to-enterprise collaboration.

In a grid-computing environment, which is a new approach to provide virtualized infrastructure, enabling people to utilize computing resources ubiquitously as utilities, CAD systems can become service providers and are available through networks in a pay-per-use fashion, in contrast to today’s buy-and-own way. A thin-client modeling environment can reduce the cost of using CAD services.

Intense human involvement is a challenge to automate the geometry creation process. Usually as the first step of design implementation—geometry creation—heavily depends on the engineers’ skills of using CAD tools. In contrast, some other design processes, such as data translation, mesh model generation, finite element analysis and simulation, and process planning, can be done in batch mode with little human intervention. Batch mode processing can increase throughput of tools and reduce the cost of service providers. It also reduces human errors and enables better design data management and knowledge reuse.

Automation of the geometry creation process will enable the geometric modeling process to be easily incorporated into a distributed CAD environment such that the work load of the client and communication channel can both be reduced. It will enable an integrated automation loop of CAD, CAE (computer-aided engineering), and optimization in design alternative evaluation. In this paper, we propose a new geometry generation mechanism—document-driven design (DDD)—for batch mode feature-based geometric modeling considering ease of communication and re-

Copyright © 2006 by ASME

Journal of Computing and Information Science in Engineering
JUNE 2006, Vol. 6 / 127
Fig. 1 Comparison between binary relation in the traditional model and a trinary relation in the semantic model: (a) Binary relations capture semantics implicitly as aggregation and association in ER-type data models, and (b) trinary relations explicitly represent semantics of constraints and design intent with good extensibility.
is an environment for multiple users to edit a shared solid object over the Web synchronously through CSG models. NETFEATURE [37,38] includes Web-enabled feature modeling clients, neutral feature model servers, and database managers. Agents are defined on the server side to serve clients for feature modeling by means of CORBA protocols. MUG [39–41] is a multiuser environment for collaborative conceptual design and shape modeling. Users are able to exchange design semantics and modify the same geometric model synchronously. WEBSPIFF [42,43] is a Web-based collaborative feature modeling system that supports interactive feature editing. Parametric representation of features is used for direct manipulation and communication. CADDAC [44,45] has a three-tier architecture, and command objects are transmitted between client and database to keep the consistency of local and master models. Li et al. [46] developed a client/server modeling framework based on B-Rep representation. A face-based feature differentiation method is used to support interactive feature editing. COLLFEATURE [47] supports nonlock multiuser feature editing. Li et al. [48] developed a neutral feature operation mapping method for collaboration of heterogeneous systems.

The above research only considers traditional interactive model construction. Batch mode feature-based modeling offers several benefits, including reduced human intervention, improved performance in distributed environments, ease of design alternative evaluation and reuse, and increased system throughput and utilization. As the distribution extensiveness of design activities increases, modeling mechanisms for complex models with ease of communication become important. The proposed DDD mechanism is to support lightweight CAD geometry construction in a service-oriented architecture with thin clients. A semantic feature model is developed to represent multilevel design intent, prevent semantics loss, and enhance data interoperability.

Semantic Feature Model

Fig. 2 Semantic richness is associated with information loss during data transformation

Fig. 3 Two levels of design intent, informative and communicative, need to be captured in semantic model: (a) solid model of anchor, (b) informative design intent is the abstract intention in the plan, (c) communicative design intent is the intention manifested during the implementation, and (d) semantic model represents design intent explicitly with subject-predicate-object triples

3 Semantic Feature Model

The semantic feature model intends to capture more design intent by providing an extensible modeling method to represent feature semantics. The fundamental difference between semantic modeling and traditional modeling methods is that traditional models represent relations between entities using binary relations, whereas the semantic model uses trinary relations. The traditional binary relations of ER-like data modeling simply model most relations as aggregation, which represents “is-part-of” relationships, and association, which represents “is-related-to” relationships. In contrast, the semantic model represents relations as subject-predicate-object triples, which explicitly capture semantics in an extensible way. The difference is illustrated in Fig. 1. In Fig. 1(a), feature relations are captured by binary aggregation and association in an EXPRESS-G diagram. In Fig. 1(b), different types of arcs represent the predicates of semantic triples, explicitly.

To be more precise, if \( E \) is a set of entities and \( R = E \times E \) is a set of relations, the semantics of a semantic feature \( f \) can be defined as \( m(f) = \{(s,p,o)\} \), where \( s, o \in E, p \in R \). For each statement, \( s \) is the subject, \( p \) is the predicate, and \( o \) is the object. The traditional feature models with binary relations only represent a subset of semantic feature models in which \( m(f) = \{ (s,p',o) \} \) and \( p' \in \{ \text{aggregation}, \text{association} \} \).

Semantic feature modeling needs to consider interoperability and extensibility. It needs to support dynamic schema evolution to capture new or evolving types of semantic information and be simple to use and lightweight. The model should not make assumptions about the semantics of the metadata. It needs to be
platform independent, provide interoperability between applications that manage and exchange metadata, and support well-formed relations for construction and query. Semantics is also local and context dependent. It should not be coded with special syntax in a tightly coupled way. Static models cannot keep pace as new requirements arrive. The semantic feature model includes three aspects for interoperability and extensibility. Intent representation is the basic requirement of feature modeling. Semantic relation representation is the essence of extensibility to represent the open set of engineering semantics, and semantics interpretation derives new semantic relations from existing ones to ensure semantic completeness within one domain.

3.1 Design Intent Representation. Semantic feature modeling is able to capture more semantics with extensible trinary relations so as to improve interoperability between different system domains. As illustrated in Fig. 2, traditional CAD data interoperability problem is resolved based on the neutral geometry model (e.g., initial graphics exchange specification (IGES) and STEP). Information loss occurs when data are translated into languages or formats that have less expressible semantics. The semantic feature model intends to capture design semantics in an extensible way. Data interoperability is improved by modeling with richer semantics. A multilevel modeling structure also increases the transparency between feature definition and feature evaluation.

There are two levels of design intent: informative and communicative. Informative design intent is the abstract intention in the plan and contains the meaning of design. Communicative design intent is manifested during the implementation and includes the meaning of designer. A semantic feature model can specify two levels of intent with properly defined feature schema. Capturing design intent requires extensible methods to represent semantics. As illustrated in Fig. 3, two levels of design intent can be captured with extensible predicates.

The semantic feature model separates implicit or intentional features from explicit or geometric features. It is important to represent two categories of features independently so that feature specification can be both procedural and declarative. High-level informational intent is in the nature of specification, whereas low-level communicative intent is more related to operation. The semantic feature model for DDD intends to migrate the way of modeling features from traditional operation oriented toward specification oriented.

Fig. 5 Membership schema can be used in feature mapping between different domains: (a) definition of feature rib in SOLIDEDGE®, which supports finite thickness extension, and (b) definition of feature rib in pro/ENGINEER®, which does not support finite thickness extension. Extra feature cut may be needed to generated the geometry of (a)

Fig. 6 Semantic interpretation helps to reduce ambiguity: (a) type I ambiguity of semantics – Different combinations of semantic features can generate the same geometry, and (b) type II ambiguity of semantics – Different geometry is created from the same semantic feature. Small variation of the parameter d causes topological differences in systems, such as SOLIDEDGE and pro/ENGINEER.
3.2 Semantic Relations. The semantic relation is the predicate in the semantic triple. The essence of flexibility and extensibility of the semantic feature model is the semantic relation between features, which, in turn, provides systematic approach for information retrieval. Basic semantic relations include static aggregation, generalization, association, and instantiation; hierarchical name spaces, which delineate contexts of semantics; membership relations, which express metalevel basic meanings of static associations; geometric relations that specify spatial association in Euclidean space; Boolean relations that specify the spatial occupation in Euclidean space; and temporal relations that capture the chronological dependency of feature evaluation.

3.2.1 Membership Schema. The membership schema is the semantic relation’s vocabulary description language for feature classes. The membership schema defines properties that are used to specify classes. The associated class relations of inheritance and instantiation are also defined. The membership schema diagram in Fig. 4 shows the scope of the schema definition. In each knowledge domain the domain schema is a structured template defined by a collection of semantic categories. A semantic category is a grouping of vocabulary within a language, organizing words that are interrelated and defined by each other in various ways. A semantic class is words that share common semantic properties or attributes.

Membership relations are metalevel relations between features, which give rules for feature creation, categorization and division, and transformation between domains. Domain ontology of feature semantics can thus be defined based on membership relations. Examples are subcategory and identical. Feature \( f_1 \) is a subcategory of feature \( f_2 \) if and only if the semantics of \( f_1 \) infers the semantics of \( f_2 \), denoted as \( m(f_1) \subseteq m(f_2) \). \( f_1 \) and \( f_2 \) are identical if \( m(f_1) = m(f_2) \). However, this universal requirement usually is too rigid for domain ontology mapping. If a semantic difference between \( m(f_1) \) and \( m(f_2) \) is defined as \( m(f_1) \setminus m(f_2) = \{(s,p,o) | (s,p,o) \in m(f_1) \land (s,p,o) \notin m(f_2)\} \), and a domain-specific semantic zero \( e \) in domain \( D \) is defined such that \( \forall f \in D, e \subseteq m(f) \), features \( f_1 \) and \( f_2 \) is identical if and only if \( m(f_1) \setminus m(f_2) \subseteq e \) and \( m(f_2) \setminus m(f_1) \subseteq e \). Extensibility is the prerequisite for membership schema because no standard cognitive notions for particular domains exist and conceptualization of terms varies in people’s perception.

The membership schema can be used in feature mapping across domains. The definitions of features are different from CAD to CAD, from CAD to CAPP, and between other systems. The mapping process can be conducted based on membership schemata. For example, the definitions of the form feature rib are different in two CAD systems, as shown in Fig. 5. Establishing mapping between two features is necessary for interoperable data exchange. In schema models, semantic mapping can be based on graph topology, special relationships, and value types. Determining the identical relation between two rib features is the process of checking the similarity or isomorphism of two schema models. Relations between ontology domains, thus, can be established.

3.2.2 Geometric Relations. Geometric relations specify the various spatial associations in Euclidean space. These relations are constraints that dynamically change the connections between feature and entities. Geometric relations specify spatial relationships in intentional features as well as in evaluated features.

3.2.3 Boolean Relations. Union, intersect, and subtract are basic Boolean operations performed during feature evaluation. A Boolean relation between features is one of the significant relations as well as one of the major problem sources in current feature-based modeling, such as naming persistency. The noncommutative property of subtract makes feature evaluation sequence dependent.
3.2.4 Temporal Relations. Temporal relations explicitly specify the chronological dependency between features as informative intent, which include precede, follow, co-occur, and independent. Temporal relations capture design history and ensure causal consistency of feature evaluation. Temporal relations are needed to complement the noncommutative property of the Boolean relation subtract.

3.2.5 Compound Relations. A compound relation allows complex features to be constructed based on basic feature definitions. Complex, but more precise semantics is needed based on the fact that compound phrases are able to express delicate meanings that are not easy to infer from the meanings of its individual parts in natural languages. For example, semantics of “white collar” is not just the intersection of semantics between “white” and “collar.” New semantics in addition to the semantics from the basic elements is generated in a compound feature. Compound relations include adjective and substantive. An adjective compound is to qualify another feature and cannot exist independently, such as countersink, Philips head, and trapezoidal runner. A substantive compound can exist independently as a complete part, such as button head rivet, helical spring lock washer, and square neck bolt. Domain-specific features can be defined with compounds, and domain semantics structure can be built based on compound relations.

3.3 Semantics Interpretation and Data Exchange. Semantics interpretation is the process of transforming a general descriptive requirement from or to a more specific system-dependent formal semantic model. Interpretation needs to manage possible one-to-many mappings. Two examples of semantics ambiguity are shown in Fig. 6. As illustrated in Fig. 6(a), one geometric model could be generated with different feature constructs type I ambiguity. The combination of low-level semantic features depends on user preference and construction sequence. In Fig. 6(b), one semantic feature can also create different geometric models with uncertain parameters caused by reference vagueness and numerical rounding errors in different systems type II ambiguity. Parameter modification of a feature could affect the features that have reference dependency on it. Different B-Rep models may be evaluated in different systems. Type I ambiguity is a planning problem, type II ambiguity is usually treated as naming persistency and model robustness problem.

3.3.1 Semantics Composition and Decomposition. A hierarchical decomposition approach can be taken to accommodate type I ambiguity. The purpose of systematic decomposition is to rationalize the design decision-making process such that arbitrary selection of semantics is avoided. Design intent needs to be captured with multiple resolutions. Based on compound relations, semantic features are constructed hierarchically. Thus, semantics can be referred to with different levels of detail. Semantics inference derives new semantics from an existing one based on axioms and rules. The feature composition process is described briefly as follows. For some adjective compound features \( A \subseteq ACF \) and substantive compound features \( \{A, B, C, \ldots \} \subseteq SCF \); if two noncommunicative composition operators are defined as \( \otimes : ACF \times ACF \rightarrow ACF \) and \( \oplus : SCF \times ACF \rightarrow SCF \), the feature composi-
is the process in which new compound features are created with the two composition operators. Examples are $A \oplus B = C$. A different way to create $C$ is that $a \oplus b = c$ and $A \oplus c = C$. Note that only one substantive compound feature is created during the composition at any time. The associated planning problem to create $A$ is to find an $X \oplus Y \oplus Z \oplus \cdots = A$. This includes the selection of both features and composition sequence.

Multiresolution intent capturing can be achieved by feature representation with different levels of details. Establishing common semantic features between system domains is required to build the bridge. Figure 7 illustrates the algorithm of searching common compound features in order to exchange feature information between two CAD domains. Identical features are searched and generated from domain-specific features based on domain rules. Once a common substantive compound feature is established, common adjective compound features can be searched further. As a result, new compound features may be defined. These high-level and commonly agreed compound features then are used for information exchange between domains. Cross-domain semantics without domain-specific details is essential to data interoperability.

Fig. 11 Feature representation and reasoning with RDF/XML documentation: (a) informational intent oriented high-level features and (b) communicative intent oriented low-level features.

3.3.2 Semantics Simplification. Semantics simplification is the process of simplifying feature dependency thus reducing type II ambiguity. The depth of feature dependency trees should be minimized during the process. Based on the continuity of geometry and the principle of semantic identification (ID) [49], stable and persistent geometric entities need to be chosen as references whenever possible. As illustrated in Fig. 8, the roots of dependency tree usually are datum planes \( x \), \( y \), and \( z \). By introducing datum features, such as planes, curves, and points, as references based on datum planes \( x \), \( y \), and \( z \), the maximum depth of the tree can be reduced to 2. Semantic equivalence relations allow for multiple ways of datum selection.

Simplified feature semantics enables history-independent modeling for global form features (e.g., extrusion, hole, cut, and loft) in which only global references are needed. In contrast, local form features (e.g., chamfer, fillet, rib, and pattern) require local references to other features. The depth of dependency trees can be reduced up to 3 if local features are involved.

In summary, the interpretation process extracts and reorganizes feature semantics when semantics is transformed from or to system-dependent feature models, during which traditional feature models are derived based on semantic compound feature models. The geometry-oriented deduction inevitably loses some design intent. The main task here is not preventing information loss. Instead, accuracy of the derived data models is the major challenge. Derivation rules need to be designed to reduce ambiguity and uncertainty of interpretation and provide robust results. This is also related to semantic relation definition in specific domains.

With complete and multilevel feature construction information, the semantic feature model with intent and relation can be represented in single or multiple documents. Document-based design interaction between client and server can be achieved simply through document processing in a distributed CAD environment.

4 Documentation of Semantic Model

Electronic document that records semantic model can be in any format. To facilitate interoperability, open standards, such as the resource description framework (RDF) / extensible markup language (XML) [50], with commonly agreed schemata are desirable, especially with the availability of low-cost parsing tools. While XML provides syntax markup, RDF enables semantics-level markup. Based on the XML syntax, RDF is a general language for representing information on the Web. In a collaborative design environment, semantic entities and relations may be located in a distributed fashion. With the RDF/XML syntax, entities and relations can be identified and linked over the Web. Feature-based geometric modeling can become a Web-based service.

4.1 RDFS for Membership Schema. RDF schema (RDFS) is RDF’s vocabulary description language used to specify domain kinds and terms. It helps to construct the structure of membership schema. The RDFS class and property system is similar to the type systems of object-oriented programming languages, such as Java. RDF differs from many such systems in that instead of defining a class in terms of the properties its instances may have, the RDFS describes properties in terms of the classes of resource to which they apply using domain and range. For example, while a classical object-oriented system might typically define a feature class Sketch with an attribute called Direction of type Vector, a Direction property has a domain of Sketch and a range of Vector in RDFS definition. With this approach, it is easy to subsequently define additional properties with a domain of Sketch or a range of...
Vector without the need to redefine the original description of these classes. This property-centric approach enhances the extensibility of the RDF. Figure 9 shows an example of RDFS representation of the membership schema in Fig. 5.

4.2 RDF for Semantic Feature Model. RDF provides a generic data format that enables Web-based intelligent information modeling, which allows for interoperability of data, machine understandable semantics for metadata, uniform query for resource discovery other than traditional text search, and integrated inference for decision making. As a standard for serializing objects, RDF facilitates document-driven processes in a Web environment.

In general, as design migrates from abstract specification to concrete feature construction, the semantics of design is enriched gradually with reasoning. Being an important part of design knowledge representation, the semantics of features can be modeled in documents such that it is machine processible. Rule-based inference engines can be used to automate the evolvement of semantics. As illustrated in Fig. 10, started from the fundamental requirement of a design or functional specification $P^0$, the compound feature is decomposed step by step toward system-specific feature construct. Based on rules, an inference engine can generate a new RDF document with richer semantics $m_i$ from the $i$th level RDF document with semantics of $m(P_i)$. Then the $i+1$th...
level RDF document with semantics of \( m(P^{+1}) \) is created with the semantic difference between \( m^f \) and \( m(P) \). The original \( m(P) \) is not necessary for the system to generate geometry. Nevertheless, to retain the original design intent, it is desirable to keep the associations among different RDF documents.

In practice, design reuse and data exchange are document archiving and sharing, and the compound feature decomposition is a process of document processing. As shown in the example of Fig. 11, from abstract to concrete, high-level features of the flange in a RDF document are replaced by low-level features systematically based on inference rules in separate documents, which are specified with the generic premise-conclusion rule syntax used in some standard RDF tools, such as Jena. Rules at different levels can also be combined and the reasoning process is shortened. While semantics is enriched as the feature model goes to detailed levels, informative intent is biased or lost as the semantics is gradually expressed by communicative intent.

The top-down generic semantics decomposition needs to be supplemented with a bottom-up domain feature composition process in order to accurately generate geometric model. Documents that define system-specific features can be created and archived separately. They are linked to higher level RDF documents. During the document processing, if semantic features are detailed enough to refer to system-specific features, these system-specific documents are used to create geometry.

4.3 Document-Centric Interaction Model. In a document-centric client-server interaction model, service consumers interact with service providers using documents that are meant to be processed as complete information. Documents could be design contents, operation request message, or both in common XML format. Simple object access protocol (SOAP) is such a communication protocol that is particularly suitable for XML-based messaging. As illustrated in Fig. 12, the document-centric interaction model enables asynchronous CAD services in batch mode as well as other engineering services, such as model translation, analysis, and simulation. Thin clients can send documents of semantic feature models in RDF format to a CAD server over networks. The CAD server will process the requests and generate CAD models in native or standard format. The CAD models can then be returned to clients. During the model generation, as the primary service, semantic features defined at remote repositories may be referred by the feature model from the client. Transparent to clients, new RDF resources may be allocated and used by the CAD server as secondary services.

Different from current Web document links, which only provide simple references for download at the syntax level, RDF provides semantic links such that meaningful information about resources can be obtained and intelligent Web services can be built.

5 Implementation

The document-driven geometric modeling mechanism based on semantic feature model is tested within the research testbed called PEGASUS at our research center. PEGASUS is a service-oriented distributed e-design system, which is to test concepts, functions, and interoperability of research prototypes as well as commercial software for collaborative design.

5.1 Service-Oriented Architecture. Service-oriented archi-
I think you may have mixed up the order of the pages. Let me retell you what I think you are referring to.

5.2 Document Processing. FIPER provides common and standard interfaces for interaction among tools as well as a process model to represent design process in conjunction with product data. Existing tools can be easily integrated in the service supply chain. At the server side, a FIPER process model is defined, which includes tasks of a document processor and a CAD service provider. The FIPER process model defines functional components for a task and their execution sequence. It also defines data flow between components in the task, as shown in Fig. 14.

An overview of the DDD system is shown in Fig. 15. The document processor is developed based on Jena [51]. Jena is an open-source RDF Java toolkit for building semantic Web applications. It provides application programing interface (API) for processing RDF and RDFS, including a generic rule-based inference engine. PRO/ENGINEER® WILDFIRE 2.0 is integrated in the process model to provide CAD services as a SIMCODE component. Based on PRO/TOOLKIT® APIs, a DDD driver for PRO/ENGINEER is developed to process incoming feature documents and generate geometric models. At the client side, the process model is accessible to thin clients with the FIPER WEBTOP Web service. Service transactions can be initiated simply through Web browsers.

The DDD mechanism enables batch mode geometric model construction based on documents that contain specifications. As illustrated in Fig. 16, a client submits documents of generic semantic features and two-dimensional (2D) sketch as the input context alone with a FIPER process model to the server. During the FIPER model execution, the inference engine generates system-specific semantic features as one or more documents based on the inputs of features and rules. These feature documents then are fed into the DDD driver of PRO/ENGINEER along with the sketch. Different three-dimensional (3D) models can be created with combinations of feature documents. Figure 17 shows how a crankshaft model is built with the DDD mechanism. After services are published at the FIPER application control system (ACS), the FIPER station can direct service requests from ACS to the service provider PRO/ENGINEER. The FIPER SIMCODE invokes PRO/ENGINEER, and the sketch document is read into PRO/ENGINEER automatically. The selection of document driven option of DDD driver will allow it to create features one by one with each feature defined in one XML document. The client can request the DDD service with a simple Web browser. The DDD mechanism supports loosely coupled and asynchronous model generation as well as lightweight design data management and access, which enables thin-client-oriented distributed CAD services. Users can control the content of documents including the FIPER process model, 2D sketch specification, semantic feature model in RDF/XML, and inference rules.

6 Conclusion
This paper presents a new feature-based modeling mechanism—document-driven design—to enable batch mode geometry construction for distributed CAD systems. This mechanism is to support loosely coupled lightweight CAD geometry generation in a service-oriented architecture with thin clients. A semantic feature model for document-driven design is developed to capture informative and communicative design intent. Feature semantics is explicitly represented as trinary relation, which provides good extensibility and prevents semantics loss. Data interoperability between domains is enhanced by schema mapping and multiresolution semantics. Semantic feature models are represented in documents with standard RDF/XML syntax such that document processing and reasoning can be easily implemented. This mechanism aims to enable asynchronous communication in distributed CAD environments with ease of design alternative evaluation and reuse, reduced human errors, and improved system throughput and utilization.

References


